



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/870,621	05/31/2001	Scott J. Broussard	AUS920010263US1	1785
35617 7590 03/31/2008 DAFFER MCDANIEL LLP P.O. BOX 684908 AUSTIN, TX 78768				
EXAMINER				
BONSHOCK, DENNIS G				
ART UNIT		PAPER NUMBER		
2173				
MAIL DATE		DELIVERY MODE		
03/31/2008		PAPER		

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte SCOTT J. BROUSSARD

Appeal 2007-2696
Application 09/870,621
Technology Center 2100

Decided: March 31, 2008

Before JAMES D. THOMAS, JOSEPH L. DIXON, and
LANCE LEONARD BARRY, *Administrative Patent Judges*.

DIXON, *Administrative Patent Judge*.

DECISION ON APPEAL

This is a decision on appeal under 35 U.S.C. § 134 from the Examiner's final rejection of claims 1-25. We have jurisdiction under 35 U.S.C. § 6(b).

We REVERSE.

BACKGROUND

Appellant's invention relates to combining the functionality of multiple text controls in a graphical user interface (Spec. 1). The present invention remedies select situations in the Swing interface which remedies situations in the Abstract Windowing Toolkit (AWT) of Java.

In a particular aspect of the claimed invention, the system of software components (172, 174 and 176, Fig. 17) provides a unique mode-switching capability, which permits two Swing objects (e.g., the JTextField and JPasswordField components) to alternate as replacements for an AWT object (e.g., the TextField component), depending on the manner in which the AWT object is being used by the application program. In one example, selection between the first and second proxy components may depend on the status of an echo character, which specifies whether entered text should be displayed as masked (e.g., "*****") or unmasked (e.g., "hello") text. (See, Specification -- page 39, lines 10-14). If a legacy application (APP 28, Fig. 1) is using an AWT TextField (170, Fig. 17) without password protection, no echo character is set. This causes the peer component (JTextFieldPeer 172) to select the first proxy component (JTextFieldProxy 174), which creates a new graphics resource component (e.g., a Swing JTextField component) for displaying the AWT TextField object without password protection. (See, Specification -- page 39, lines 14-16). If an echo character

is set, however, the peer component (JTextFieldPeer 172) selects the second proxy component (JPasswordFieldProxy 176), which creates a new graphics resource component (e.g., a Swing JPasswordField component) for displaying the AWT TextField object with password protection. (See, Specification -- page 39, lines 16-19). In either case, the newly created graphics resource component displays the object in a manner, which is independent of the operating system. (See, Specification -- page 39, lines 19-25). (Br. 3-4). An understanding of the invention can be derived from a reading of exemplary claim 1, which is reproduced below.

1. A system of software components adapted to display an object created by an application program running under an operating system, wherein the system of software components comprises:

a first proxy component;

a second proxy component; and

a peer component for selecting either the first proxy component or the second proxy component, depending on a mode of use of the object, wherein the selection can be made during runtime, and wherein after the proxy component is selected, the selected proxy component dynamically creates a new graphics resource component for displaying the object, such that the appearance of the displayed object is substantially independent of the operating system.

PRIOR ART

The prior art references of record relied upon by the Examiner in rejecting the appealed claims are:

Fults

US 5,327,529

Jul. 5, 1994

Sun Microsystems, Java Platform 1.2 Beta 4 API Specification: Class JPasswordField and Class JTextField, 1993-1998, 1, (hereinafter Java)

Sun Microsystems, The Swing Connection, 2/98, volume 3, no.4, swing version 1.0, (hereinafter Java (specifically IS)).

WinZip Computing Inc., WinZip 8.0, 1991-2000, attached pages, (hereinafter WinZip)

REJECTIONS

Claims 1-25 are rejected under 35 U.S.C. § 103(a) as being unpatentable over WinZip, Java, and Fults. (We note that the Examiner has not applied the Java IS reference against the claims, but does refer to it in some rejections; it has not been considered as part of the rejection).

Rather than reiterate the conflicting viewpoints advanced by the Examiner and Appellant regarding the above-noted rejection, we make reference to the Examiner's Answer (mailed March 7, 2007) for the reasoning in support of the rejections, and to Appellant's Brief (filed November 15, 2006) for the arguments thereagainst.

OPINION

In reaching our decision in this appeal, we have given careful consideration to Appellant's Specification and claims, to the applied prior art references, and to the respective positions articulated by Appellant and the

Examiner. As a consequence of our review, we make the determinations that follow.

35 U.S.C. § 103

In rejecting claims under 35 U.S.C. § 103, it is incumbent upon the Examiner to establish a factual basis to support the legal conclusion of obviousness. *See In re Fine*, 837 F.2d 1071, 1073 (Fed. Cir. 1988). In so doing, the Examiner must make the factual determinations set forth in *Graham v. John Deere Co.*, 383 U.S. 1, 17 (1966). “[T]he Examiner bears the initial burden, on review of the prior art or on any other ground, of presenting a *prima facie* case of unpatentability.” *In re Oetiker*, 977 F.2d 1443, 1445 (Fed. Cir. 1992). Furthermore, “‘there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness’ . . . [H]owever, the analysis need not seek out precise teachings directed to the specific subject matter of the challenged claim, for a court can take account of the inferences and creative steps that a person of ordinary skill in the art would employ.” *KSR Int’l Co. v. Teleflex Inc.*, 127 S. Ct. 1727, 1741 (2007)(quoting *In re Kahn*, 441 F.3d 977, 988 (Fed. Cir. 2006)).

Appellant argues that the present claimed invention provides a unique system of software components that enables an application program (Java application) to be truly portable across all operating systems (OS) platforms by providing various means for maintaining the “look and feel” of the application program. (App. Br. 6). Appellant argues that the Java applications programs utilize a platform-dependent application program

interface (API), commonly known as the Abstract Windowing Toolkit (AWT), to produce what are called heavyweight software components that are written using native code (i.e., instructions specific to the particular Operating System). In that the Look and Feel of a GUI refers to such things as the appearance color in behavior of the interface and there are limitations in the AWT that prohibit true portability of the Java application programs, especially in terms of the Look and Feel of the application programs (App. Br. 6-7).

Appellant argues that the program “Swing” was developed to contain no native code, but unfortunately these lightweight software components cannot completely eliminate the platform dependency of Java applications that use the underlying AWT (App. Br. 7). The present patent application involves the mixing of Swing and AWT components within a given application program interface in an attempt to replace the dual mode, AWT text field components within the lightweight Swing counterpart. (*Id.*) Appellant argues that in order to utilize Swing components, while maintaining behavioral compatibility with the Java application program, the present invention recognizes that the appropriate Swing components must be dynamically re-created and substituted for the AWT components each time the mode of use of the object changes which is in direct contrast to the manner in which Swing components are normally created, which is during the initial construction of the peer components. (*Id.* at 8).

Appellant argues that the presently claimed system of software components provides a unique mode-switching capability that allows two

Swing software components to alternate as replacements for an AWT software component depending on the manner in which the object is being used. (*Id.*). Appellant argues at pages 8-9 of the Brief that the mode switching capability includes a first proxy component, a second proxy component and a peer component and that the proxy components are a set of executable code that when executed translates the method calls from the AWT components to the appropriate Swing components, and translates the event calls from a Swing components to the AWT components containing the Swing component. Appellant argues that the peer component is used for selecting either the first proxy component or the second proxy component, depending on the mode of use of the object. (*Id.*)

Appellant argues the none of the cited art either separately or in combination, provides motivation to teach or suggest a system of software components for selecting, during real time, a first proxy component or a second proxy component for use in displaying an object, where the selection is dependent on a mode of use of the object and where the appearance of the displayed object is substantially independent of the operating system. (*Id.*) Appellant argues, at pages 9-14 of the Brief, the combination of WinZip, Java, and Fults is insufficient to teach or suggest the claimed invention. We generally agree with Appellant's analysis and do not find that the Examiner has made a persuasive initial showing of obviousness under 35 U.S.C. § 103 based upon the combination of WinZip, Java, and Fults.

In the responsive arguments section of the Answer, the Examiner mentions the Java IS reference with respect to independent claim 1, but has

not applied this reference against independent claim 1. Additionally, even if applied, we do not find that the Java IS reference would remedy the deficiencies in the base combination since the operating system independent display is not actually independent of the operating system at all times as noted on page 5 of Java IS reference with a discussion of the Swing interface. Java IS discloses that the Swing interface does not sit atop all AWT components that existed prior to the introduction of Swing. Hence, the Examiner's reliance upon the Java IS reference to teach that Swing is completely independent of the operating system is misplaced.

From our review of these teachings, the Examiner's reliance upon the WinZip reference screen shots to be deficient as to show the process of how the end result was produced. Additionally, the Java reference while teaching Swing components, does not show that the same Look-and-Feel is produced no matter what operating system they are implemented on, as asserted by the Examiner at page 15 of the Answer.

We find the Examiner's rejection to additionally be based upon hindsight reconstruction in an attempt to reconstruct the claimed invention after Appellant has identified a problem which has not been recognized in the prior art applied against the claims. Therefore, we do not find sufficient motivation for the combination of references as maintained by the Examiner in the rejection. Therefore, for the above noted deficiencies, we cannot sustain the rejection of independent claim 1 and its dependent claims 2-12. For the same reasons, we cannot sustain the rejection of independent claim 13 and its corresponding dependent claims 14-24 and independent claim 25.

CONCLUSION

To summarize, we have reversed the rejection of claims 1-25 under 35 U.S.C. § 103(a).

REVERSED

Appeal 2007-2696
Application 09/870,621

rwk

DAFFER MCDANIEL LLP
P.O. BOX 684908
AUSTIN, TX 78768